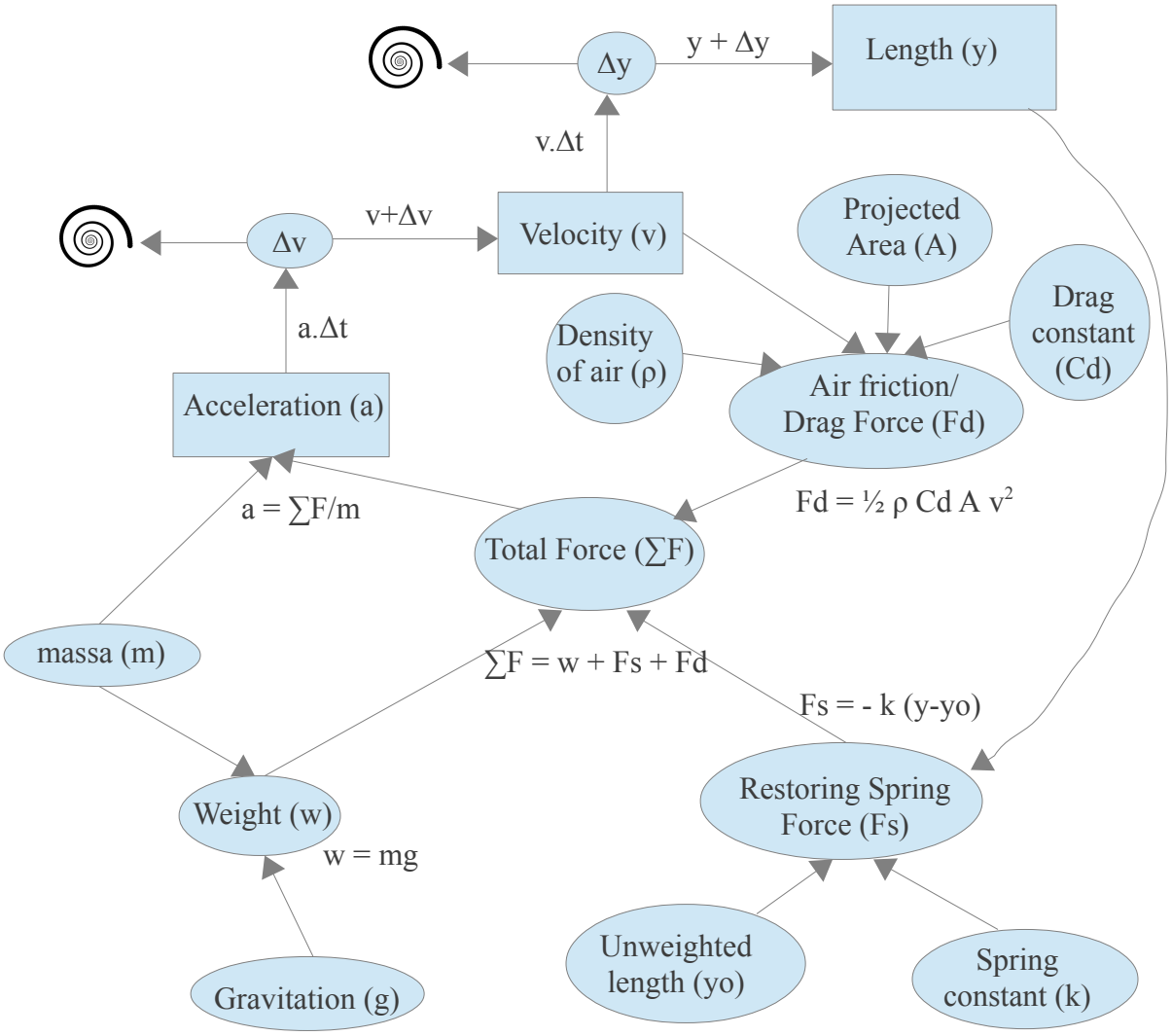
Computational Science – 2012
Tugas Pengenalan Sains Komputasi
Ridlo W. Wibowo – 20912009

Bungee-Jumping

Lengkapi, perbaiki, dan buat pseudocode.

Melengkapi desain simulasi



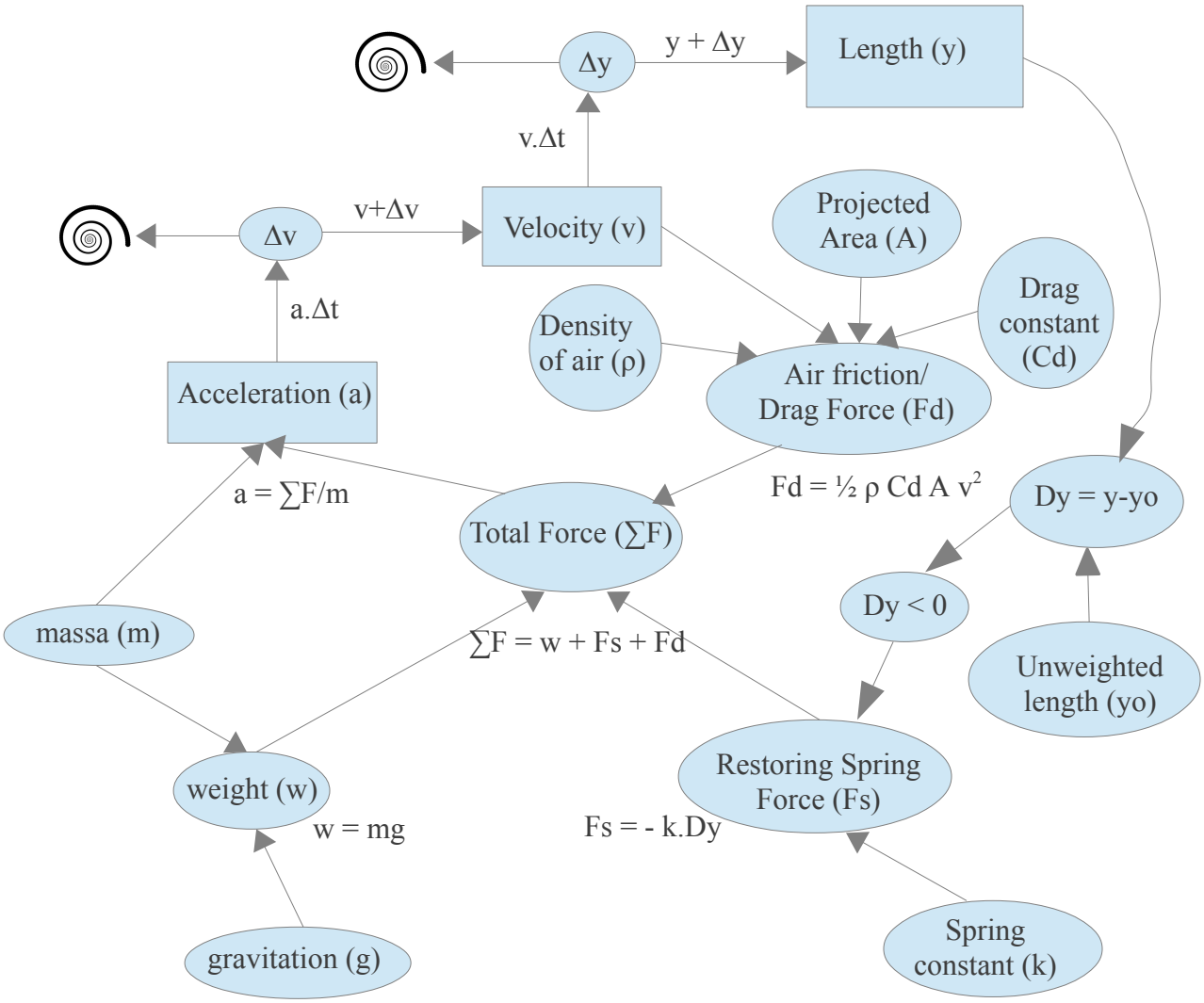
Perbaikan:

Algoritma di atas bertujuan untuk menggambarkan persamaan gerak dari seseorang yang melakukan *Bungee-Jumping*, namun terdapat beberapa kesalahan yang masih dapat diperbaiki. Kesalahan terbesar dari algoritma di atas adalah menggunakan persamaan gaya pegas saja sebagai pengganti gaya pegas tali.

Apa yang terjadi adalah ketika orang itu baru memulai *bungee-jumping* dia sudah terkena gaya pegas karena terdapat (Δy), sehingga dia tidak mengalami peristiwa jatuh bebas, namun tertarik oleh "pegas" tersebut. Begitu pula ketika orang yang melakukan bungee-jumping berada di atas (yo), dia akan tertarik ke bawah bukan jatuh bebas seperti apabila kita menggunakan "tali".

Perbaikan sederhana dapat dilakukan dengan menambahkan filter, apakah $y < yo$, jika iya maka gaya pegas ada, sedangkan jika $y > yo$ gaya pegas menjadi "nol" (dengan acuan yo positif dan orang memulai bungee-jumping tentunya di atas yo). Dengan melakukan ini maka "sifat tali" dan "pegas"nya dapat diperoleh (sehingga lebih mirip orang yang bermain *trampoline*).

Hasil perbaikan:



Pseudocode (Euler)

```
main:{
input: m, g, k, Cd, rho, A, to, tf, n, y0, yo
#massa, perc grav, konstanta pegas, drag constant(geometri & materi)
#density udara, luas permukaan gesekan, awal integrasi,
#akhir integrasi, jumlah step, tinggi awal lompatan,
#dan panjang tali (unweighted length)

dt = (tf-to)/n #time step
#inisiasi
t=to #waktu awal loncat to=0
y=y0 #ketinggian awal loncatan
v=0  #kecepatan awal loncat = 0

for i=1,n do begin
     a  = totalF(t,v,y)/m #memanggil fungsi total gaya
     v  =  v + a.dt
     y  =  y + v.dt
     print(t, a, v, y)
     t  =  t + dt
endfor
}

function totalF(t,v,y):{#fungsi total gaya, memanggil fungsi fd & fs
     w = m.g
     return fs(t,y) + fd(t,v) + w
}

function fs(t,y):{ #fungsi gaya pegas
     Dy = y-yo
     if (Dy>0){return 0 }
     else{return -k.Dy }
}

function fd(t,v):{ #fungsi gaya gesek
     konst = ½.Cd.rho.A
     gesek = konst.v^2
     if (v<0){ return gesek) #selalu berlawanan dengan arah v
     else{ return -gesek}
}
```

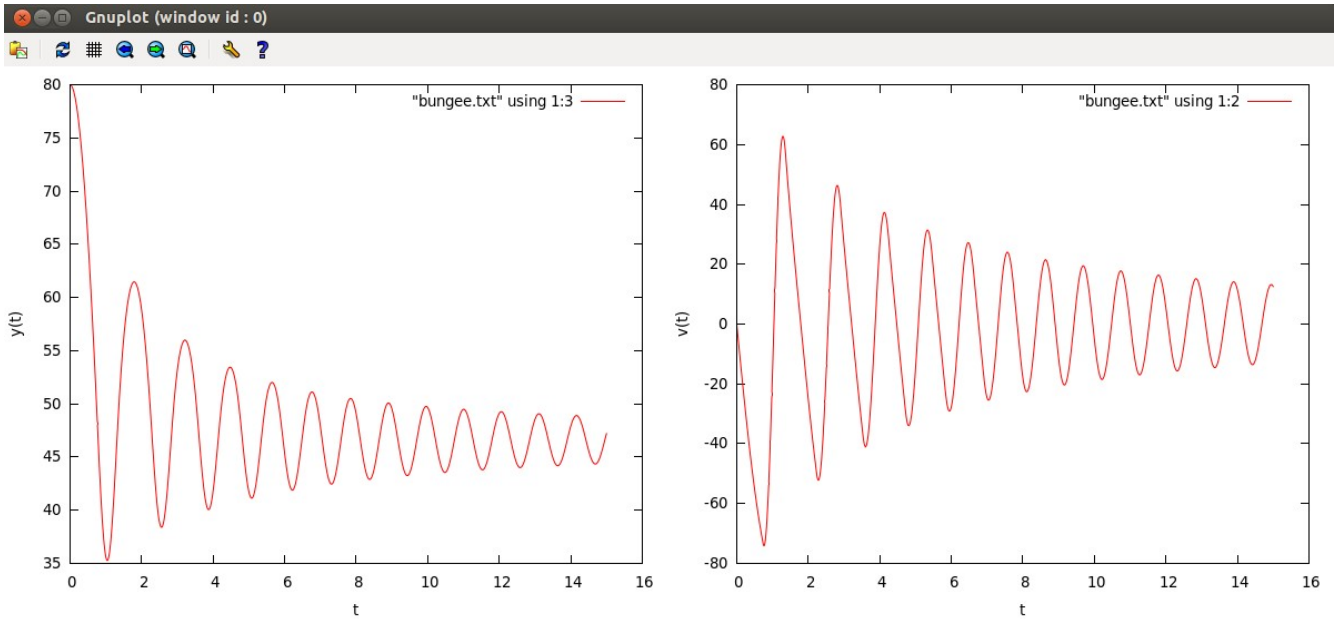
Code (Runge-Kutta 4th order)

```
/*********************************************************************/
/* bungee.cpp / trampoline                                           */
/* Copyleft (c) 2012. Ridlo W. Wibowo                                */
/*********************************************************************/

#include<iostream>
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<fstream>
using namespace std;

double Cd=0.1, rho=1.225, A=1.;
double g=9.8;
double k=200.;
double yi=80., yo=50.; // panjang tali 30 meter
double m=70.;

double totalF(double t, double v, double y);
double fd(double t, double v);
double fs(double t, double y);

int main(){
    int N = 10000, i, j;
    double ti=0.0, tf=15.;
    double vi=0.0, h;

    h = (tf-ti)/(double) N;

    double y[N], v[N], t[N];
    y[0] = yi;
    v[0] = vi;
    t[0] = ti;

    // Runge-Kutta
    for (i=1;i<=N;i++){
        double ky1, kv1, ky2, kv2, ky3, kv3, ky4, kv4;
        ky1 = h*v[i-1];
        kv1 = h*totalF(t[i-1], v[i-1], y[i-1])/m;
        ky2 = h*(v[i-1] + kv1/2.);
        kv2 = h*totalF(t[i-1] + h/2., v[i-1] + kv1/2., y[i-1] + ky1/2.)/m;
        ky3 = h*(v[i-1] + kv2/2.);
        kv3 = h*totalF(t[i-1] + h/2., v[i-1] + kv2/2., y[i-1] + ky2/2.)/m;
        ky4 = h*(v[i-1] + kv3);
        kv4 = h*totalF(t[i-1] + h, v[i-1] + kv3, y[i-1] + ky3);

        y[i] = y[i-1] + (ky1 + 2.*(ky2+ky3) + ky4)/6.;
```

```
        v[i] = v[i-1] + (kv1 + 2.*(kv2+kv3) + kv4)/6.;
        t[i] = t[i-1] + h;
    }

    ofstream outfile;
    outfile.open("bungee.txt");
    for (j=0;j<=N;j++){
        outfile << t[j] << " " << v[j] << " " << y[j] << "\n";}
    outfile.close();

    ofstream ploter;
    ploter.open("plot_bungee.in");
    ploter << "# gnuplot input file\n";
    ploter << "set terminal wxt size 1200, 500\n";
    ploter << "set multiplot layout 1,2\n";
    ploter << "set xlabel \"t\"\n";
    ploter << "set ylabel \"y(t)\"\n";
    ploter << "plot \"bungee.txt\" using 1:3 with lines\n";
    ploter << "set ylabel \"v(t)\"\n";
    ploter << "plot \"bungee.txt\" using 1:2 with lines\n";
    ploter << "unset multiplot\n";
    ploter.close();

    system("gnuplot -persist < plot_bungee.in");

    return 0;
}

double totalF(double t, double v, double y){
    double w;
    w = m*g;
    return fs(t,y) + fd(t,v) - w;
}

double fd(double t, double v){
    double konst, gesek;
    konst = 0.5*Cd*rho*A;
    gesek = konst*v*v;
    if (v < 0.){return gesek;}
    else{ return -1.*gesek;}
}

double fs(double t, double y){
    double Dy;
    Dy = y-yo;
    if (Dy > 0.){return 0;}
    else{ return -k*Dy;}
}
```

Hasil

Dengan parameter sesuai kode di atas, hasil run sebagai berikut:
Cd = 0.1 // setengah streamline {sumber: wikipedia}
rho = 1.225 kg/m^3
A = 1. m^2
g = 9.8 m/s^2
k = 200. N/m
yi = 80. // ketinggian awal jatuh bebas
yo = 50. // panjang tali 30 meter
vi = 0.0 // jatuh bebas pada saat awal (yi)
m = 70. kg
N = 10000, ti=0.0, tf=15. // integrasi hingga 15 detik pertama



Terlihat bahwa ketinggian berkurang kemudian berosilasi, begitu juga dengan kecepatan.

Menurut teori posisi orangnya (y) akan konvergen pada

$y = yo - w/k$

$y = 50 - 700/200$

$y = 46.5$ m

dan kecepatan (v) akan konvergen menuju $\rightarrow$ 0 m/s^2
Apabila simulasi dilakukan dengan tf lebih besar maka akan mencapai angka tersebut, menunjukkan simulasi menuju/konvergen ke nilai yang benar.